Computer Science

Parallel Computer Generated Holography

Stanislav V. Miroshnikov, Mathematics and Computer Science Department, Manhattan College, NY 10471 (smiroshn@student.manhattan.edu)
Jason Grigsby, Computer Science Department, Birmingham-Southern College, AL 35254 (jdgrigsb@panther.bsc.edu)
Corey Kovacs(*), Computer Science Department, DePauw University, IN 46135 (ckovacs@depauw.edu)
David E. Maharry(*), Mathematics and Computer Science Depart., Wabash College, IN 47133 (maharryd@wabash.edu)

Holography is the art or process of making or using a hologram. In computer generated holography the object is described by a set of source points that are used in formulas to calculate the intensity of the hologram image at each pixel of the imaginary film. In this work we used the cellular approach that was developed under direction of Prof. Douglas Harms at the DePauw University REU program last year. Because this approach is computationally intensive, it took a good deal of time to create a relatively small hologram. Therefore we developed a parallel solution to this problem.

We converted the algorithm to run in parallel on a Beowulf cluster consisting of 16 Pentium processors at DePauw University. Our algorithm was implemented in C++, using the LAM-MPI 2 library.

Our first approach to was to evenly divide the hologram into strips of pixels and distribute the computation for each strip between the processors. Our results indicate this is an efficient algorithm providing a speed up proportional to the number of processors. However, since this algorithm required us to store in memory the values of the intensities for each pixel at the same time, when the hologram size was large the memory required exceeded the physical memory of the nodes. This caused the system to begin disk swapping and the execution time increased dramatically.

We developed two more algorithms to circumvent this memory issue. Our new "set-strips" algorithm involved limiting the size of the strip that was processed by one node at a time. The implementation used the master node as a manager to assign strips to slave nodes and thus it was not involved in computation. Slave nodes had to calculate the values of the intensities of each pixel twice so as not to require all of them to be in memory at the same time. The last "no-strips" algorithm abandoned the idea of storing all the pixels in memory. Slaves would only keep one pixel at a time and extract information about that pixel as needed. As in the "set strips" algorithm the pixels intensities had to be calculated twice.

Timing results for all the algorithms were obtained for different numbers of source points, sizes of hologram and numbers of processors and will be presented. Our results indicate that the two algorithms were efficient solutions for holograms that exceed the physical memory of the machines. The “set-strips” algorithm was found to be a good compromise between memory and computational demands of the problem.

As a final step in our project we built a 5-node Beowulf cluster with Macintosh G3 processors running the Linux operating system. We successfully compiled and ran our programs on the cluster without modifying the code. Our results include timing comparisons between the two clusters.

Acknowledgement: NSF REU grant number EIA-9911626 supported this work.